robotron

# Sprachbeschreibung der Makrobefehlssprache MABS 1520 SUBSET

# Betriebssystem BROS

## Vorwort:

Die vorliegende Assemblersprachbeschreibung erläutert die MABS 1520-Sprache, die bei der Programmierung von Applikationsprogrammen mit dem Gerät robotron 1715 zum Einsatz kommt. Diese Sprachbeschreibung dient nicht zum Erlernen des Programmierens.

Die Assemblersprachbeschreibung enthält die Beschreibung der Sprache. Die Beschreibung der Wirkung der Makrobefehle wird in der Schrift "System- und Programmierhandbuch für Betriebssystem BROS vorgenommen.

© VEB Robotron - Büromaschinenwerk Sömmerda 1985

Inhaltsverzeichnis


| | | Seite |
|---|---|---|
| 1. | Charakteristika von MABS 1520 (Subset) | 4 |
| 2. | Metasprache für die Syntaxdarstellung | 4 |
| 3. | Format der Sprache MABS 1520 (Subset) | 4 |
| 3.1. | Befehlszeile | 5 |
| 3.2. | Kommentarzeile | 6 |
| 3.3. | Folgezeile | 6 |
| 3.4. | Quellprogrammbeispiel | 7 |
| 4. | Sprachelemente | 7 |
| 4.1. | Zeichen | 7 |
| 4.1.1. | Alpha-numerische Zeichen | 8 |
| 4.1.2. | Hexadezimalziffern | 8 |
| 4.2. | Mnemonik | 8 |
| 4.3. | Symbol | 8 |
| 4.4. | Bezugnahme auf den Speicherplatzzuweisungszähler | 9 |
| 4.5. | Dezimalzahlen | 9 |
| 4.6. | Adreßdirektwert | 10 |
| 4.7. | Ausdrücke, Terme | 10 |
| 4.7.1. | Längenattribut | 10 |
| 4.7.1.1. | Explizite Länge | 10 |
| 4.7.1.2. | Implizite Länge | 11 |
| 4.7.2. | Adreßtypen von Ausdrücken | 11 |
| 4.7.2.1. | Adreßtyp 1 | 11 |
| 4.7.2.2. | Adreßtyp 2 | 12 |
| 4.7.2.3. | Adreßtyp 3 | 12 |
| 4.7.2.4. | Adreßtyp 4 | 12 |
| 4.7.2.5. | Adreßtyp 5 | 12 |
| 4.7.3. | Indirekte Adressierung | 12 |
| 4.8. | Direktoperanden | 13 |
| 4.8.1. | Direktoperand der Länge 1 Byte | 13 |
| 4.8.2. | Direktoperand der Länge 2 Byte | 13 |
| 4.9. | Geräteadresse | 13 |
| 4.10. | Sprungbedingung in Abhängigkeit von einem der Selektoren 0 - 7 | 13 |
| 4.11. | Spezifikation der Selektoren im K-Register | 13 |
| 4.12. | ON-Codes | 14 |
| 4.13. | ONF-Codes | 14 |
| 4.14. | Positionierung bei sequentiellen Datei-E/A-Befehlen | 14 |
| 4.15. | Positionierparameterliste | 15 |
| 4.16. | Funktionscodes | 16 |
| 4.17. | Starttastencodes | 16 |
| 4.18. | Maske | 16 |
| 4.19. | Konstantenwert | 17 |
| 5. | Beschreibung der MABS 1520 - Pseudobefehle | 18 |
| 5.1. | Beginn eines Quellprogramms (PN) | 18 |
| 5.2. | Ende eines Quellprogramms (END) | 18 |
| 5.3. | Seitenschaltung im Übersetzungsprotokoll (EJEC) | 18 |
| 5.4. | Zeilenschaltung im Übersetzungsprotokoll (SPACE) | 18 |
| 5.5. | Kennzeichnung externer Symbole (EXTRN) | 19 |

| | | |
|---|---|---|
| 5.6. | Definition symbolischer Eintrittspunkte (ENTRY) | 19 |
| 5.7. | Speicherbereichsdefinition (DS) | 21 |
| 5.8. | Konstantendefinition (DC) | 21 |
| 5.8.1. | Festkommakonstante 2 Byte (TYP F) | 22 |
| 5.8.2. | Festkommakonstante 1 Byte (TYP I) | 22 |
| 5.8.3. | BCD-Konstante 4 Byte (TYP W) | 22 |
| 5.8.4. | BCD-Konstante 8 Byte (TYP D) | 23 |
| 5.8.5. | ISO-Zeichenketten (TYP C) | 23 |
| 5.8.6. | Hexadezimalkonstante (TYP H) | 24 |
| 5.8.7. | Adresskonstante (TYP A) | 24 |
| 5.8.8. | Adresskonstante (TYP Y) | 25 |
| 5.8.9. | Adresskonstante (TYP V) | 25 |
| 5.8.10. | Adressdirektwert 2 Byte (TYP X) | 25 |
| 5.9. | Wertzuweisung für ein Symbol (EQU) | 25 |
| 5.10. | Setzen des Speicherplatzzuweisungszählers (ORG) | 26 |
| 5.11. | Definition des statischen Dateidefinitionsblockes (DDB) | 26 |
| 5.12. | Definition des dynamischen Dateidefinitionsblockes (DCB) | 29 |
| Anlage 1: | Einteilung der Makrobefehle nach der Operandenstruktur | 30 |
| Anlage 2: | Liste der Makro- und Pseudobefehle | 32 |
| Anlage 3: | Übersicht zu den Sprachelementen | 38 |

## 1. Charakteristika von MABS 1520 (Subset)

Die Sprache MABS 1520 ist eine maschinenorientierte symbolische Assemblerprogrammiersprache (MABS = Makro-Befehls-Sprache).

Die Makrobefehle sind Maschinenbefehle einer höheren Ebene (über CPU-Befehls-Ebene) und werden durch den Makrointerpreter interpretativ abgearbeitet. Weiterhin bietet MABS 1520 eine Reihe von Pseudobefehlen, die der Steuerung der Arbeit des Assemblers dienen.
Der Befehlsumfang stellt eine Untermenge (Subset) von MABS 1520 dar. Damit werden die Belange der Datenerfassung-, Buchungs- und Fakturiertechnik berücksichtigt.
Die Übersetzung in MABS 1520 geschriebener Programme erfolgt mit dem Übersetzungsprogramm ASSEM.

Symbolische Programme sind leichter lesbar und übersichtlicher als Programme im Maschinencode. Ihre Erstellungs-, Test- und Änderungszeit ist kürzer, da bestimmte Fehlerquellen vermieden werden und bei der Übersetzung formale Sprachfehler vom MABS 1520 erkannt werden.

Die sprachlichen Mittel gestatten bei symbolisch geschriebenen Programmen, verschiebliche Programme zu erstellen sowie für einzeln erstellte Programmteile, Programme und Unterprogramme die logischen Verbindungen untereinander zu definieren.

## 2. Metasprache für die Syntaxdarstellung

Zur Darstellung der Sprachsyntax wird eine Metasprache benutzt, die nach folgenden Regeln aufgebaut ist:

1. Worte der Metasprache werden klein geschrieben.
2. In der Metasprache finden folgende Zeichen mit definierter Bedeutung Anwendung:

| Zeichen der Metasprache | Bedeutung |
|---|---|
| [ ] | Ein in eckige Klammern eingeschlossenes Element der Sprache kann fehlen |
| : := | "besteht aus" |
| / oder ( ) | "/" oder "( )" sind Auswahlzeichen, d.h. sie trennen gleichberechtigte Elemente (Funktion: oder) |
| [ ] | Das in eckige Klammern eingeschlossene Element kann i mal auftreten |
| := | "ergibt sich zu" |

## 3. Format der Sprache MABS 1520 (Subset)

Ein Quellprogramm besteht aus einer Folge von Quellzeilen.

Die maximale Länge einer Quellzeile beträgt 53 Zeichen. Quellzeilen werden mit dem Editor des Assemblers ASSEM erfaßt. Der Editor generiert automatisch das Zeilenende (NL). Eine Quellzeile kann sein

- eine Befehlszeile oder
- eine Kommentarzeile oder
- eine Folgezeile (nur nach DDB-Befehlszeile!).

3.1. Befehlszeile

Eine Quellzeile, die einen symbolischen Befehl enthält, besteht aus 4 Feldern variabler Länge. Sie werden durch jeweils ein Trennzeichen voneinander getrennt:

| Markenfeld : | Operations-codefeld | Operandenfeld; | Kommentarfeld ET1 |
|---|---|---|---|

Die Quellzeile wird immer durch ET1 abgeschossen.

Markenfeld

Das Markenfeld kann maximal 5 Positionen umfassen. Es wird benutzt, um einem Befehl oder einer Konstanten bzw. einem Speicherbereich eine symbolische Adresse (Symbol) zuzuweisen. Damit kann auf diese symbolische Adresse von jeder beliebigen Programmstelle Bezug genommen werden.

Die Symbole bestehen aus 1 bis 5 Buchstaben oder Ziffern, wobei das 1. Zeichen ein Buchstabe sein muß.

Das Markenfeld wird durch das Trennzeichen ':' abgeschlossen. Das Trennzeichen folgt unmittelbar auf die letzte Stelle des Symbols. Enthält eine Quellzeile kein Symbol im Markenfeld, so ist für den Doppelpunkt ein Leerzeichen zu setzen.

Operationscodefeld (OP-Codefeld)

Das Operationscodefeld enthält die Befehlsmnemonik (Operationscode der symbolischen Anweisung). Die Makrobefehlsmnemonik besteht aus 1 bis 5 Zeichen.

Dieses Feld weist die Maschinenoperation bzw. Pseudooperation an, die auszuführen ist.

Das Operationscodefeld wird vom Operandenfeld durch ein Leerzeichen (ϕ) getrennt.

Operandenfeld

Das Operandenfeld dient der Aufnahme der Operanden (Operandenliste), die vom Typ des Befehls abhängen (siehe Operandentypen Abschnitt 4 und Gruppeneinteilung der Befehle nach Operandentypen in Anlage 1).

Das Operandenfeld endet mit ET1, wenn kein Kommentar folgt bzw. mit ';', falls Kommentar folgt.

*Kommentarfeld*

*Kommentar erleichtert dem Programmierer das Verständnis und die Wartung des Programms.*

*Ein wahlweise angefügter Kommentar hat mit einem Semikolon (';') zu beginnen und endet mit ET 1.*

*Formaler Aufbau einer Befehlszeile*

```
(      ∅        )
(   [Symbol :]  )  OP-Code  [∅ Operanden-]    [;Kommentar]  ET 1
(                )                liste
(                )
```

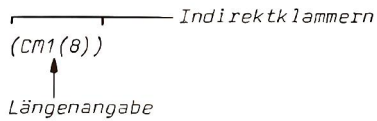
*Enthält eine Befehlszeile kein Symbol im Markenfeld, so ist ein Leerzeichen vor dem Operationscode zu setzen.*

*Beim Schreiben des Operandenfeldes eines symbolischen Befehls sind folgende syntaktische Regeln zu beachten:*

1. *Das Operandenfeld ist von OP-Codefeld durch ein Leerzeichen zu trennen.*
   *Abgeschlossen wird es durch ET 1 bzw. wenn Kommentar folgt durch ;Kommentar ET 1*
2. *Operanden untereinander werden durch Komma getrennt. Leerzeichen (∅) zwischen den Operanden sind verboten.*
3. *Leerzeichen innerhalb der Operanden sind nur bei Zeichenketten zugelassen.*
4. *Längenangaben, die zu Ausdrücken gehören, werden in runde Klammern '()' eingeschlossen.*
5. *Wird eine Indirektadressierung vorgenommen, ist der Ausdruck einschließlich einer möglichen Längenangabe in Klammern einzuschließen.*

   *z.B.*

```
┌──────────┬──── Indirektklammern
(CM1(8))
   ↑
Längenangabe
```



*3.2. Kommentarzeile*

*Eine Befehlszeile gilt als Kommentarzeile, wenn sie mit ";" beginnt.*

*Formaler Aufbau der Kommentarzeile:*

*;Kommentar ET 1*

*Die Kommentare haben auf die Übersetzung keinen Einfluß.*

*3.3. Folgezeile*

*Folgezeilen sind nur bei der Pseudoanweisung DDB erlaubt.*

*Eine Folgezeile wird erwartet, wenn eine DDB-Anweisung im Operandenfeld mit Komma (,) endet (siehe 5.11.).*

*Aufbau einer Folgezeile*

*ϕ DDB ϕ Operandenliste [,] [;Kommentar] ET 1*

*3.4. Quellprogrammbeispiel*

```
                PN     ABC
;* * * * * * * * * * * * * * * * *
;*
;* EINGABE + DRUCK NUM. WERTE
;*
;* * * * * * * * * * * * * * * * *

                SOP
MA312:          POS    1, H = 1Ø, V + 1          ;DRUCKPOSITION
                ENTP   FELD                      ;NUM. EINGABE
                PUTP   1, Z, 7, FELD             ;DRUCK
                B      MA312
FELD:           DS     4
```

*4. Sprachelemente*

*4.1. Zeichen*

*Für MABS 1520 gelten die Codefestlegungen nach RGW 956-76, 7-Bit-Code.*

*Einige Zeichen haben eine besondere Bedeutung in der MABS 1520-Sprache:*

*Elemente der Sprache mit besonderer Bedeutung:*

| Zeichen | Bedeutung |
|---|---|
| : | Im Markenfeld in Verbindung mit einem Symbol als Kennzeichnung der Marke und damit Trennzeichen zwischen Marken- und OP-Codefeld |
| * | Im Operandenfeld innerhalb eines Ausdruckes gibt * den laufenden Stand des Speicherplatzzuweisungszählers an |
| + - | Operatoren in Ausdrücken (s. Abschnitt 4.7.) |
| , (Komma) | Trennungszeichen für Ausdrücke in der Operandenliste des Operandenfeldes |
| ; | 1. als erstes Zeichen einer Quellzeile kennzeichnet es eine Kommentarzeile<br>2. wird es im Kommentarfeld angegeben, dient es als Trennungszeichen zwischen Operanden und Kommentarfeld |
| ( ) | Doppelte Bedeutung:<br>1. Kennzeichnung der expliziten Länge eines Ausdrucks |

| Zeichen | Bedeutung |
|---|---|
| | 2. Kennzeichnung der Indirektadressierung (adr) (adr (Längenattribut)) |
| = | Spezifikation eines Parameters |
| ' (Apostroph) | Kennzeichnung einer Zeichenkette |

### 4.1.1. Alpha-numerische Zeichen ($c_i$)

Es sind alle alpha-numerischen Zeichen entsprechend RGW 356-76, 7-Bit-Code, zugelassen (Spalten 2-7 der Codetabelle).

$$c \in \{\circledast, A, \ldots, Z, \emptyset, \ldots 9, \not{b}, ! \ldots\}$$

### 4.1.2. Hexadezimalziffern ($h_i$)

Eine Hexadezimalziffer (auch Hexaziffer) ist eines der nachfolgenden Zeichen:

$$h \in \{\emptyset, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

## 4.2. Mnemoniks

Die symbolischen Abkürzungen der Makrobefehle und Pseudobefehle werden als Mnemoniks (bzw. Befehlsmnemoniks) bezeichnet. (Mnemoniks der Makrobefehle können 1 bis 5 Zeichen, Mnemoniks der Pseudobefehle 2 bis 5 Zeichen umfassen.)

Die von MABS 1520 (Subset) für die Makrobefehle zugelassenen Mnemoniks werden in Anlage 2 angegeben. Eine Beschreibung der Pseudobefehle und die Angabe ihrer Mnemoniks erfolgt unter Pkt. 5.

## 4.3. Symbol

Ein Symbol besteht aus maximal 5 Zeichen, beginnend mit einem Buchstaben. Weitere Zeichen können Buchstaben oder Ziffern sein.

Sonderzeichen sind nicht zugelassen.

Symbole können sowohl im Markenfeld als auch im Operandenfeld stehen. Ein Symbol verkörpert einen Adreßwert und ein Längenattribut. Der Adreßwert eines Symbols im Markenfeld ist gleich dem aktuellen Wert des Speicherplatzzuweisungszählers. Ist der markierte Befehl (Symbol im Markenfeld) ein Makrobefehl, so ist das Längenattribut gleich 1. Ist der markierte Befehl eine Konstantendefinition oder eine Speicherplatzvereinbarung, so ist das Längenattribut gleich der Länge der Konstanten oder des reservierten Speicherbereiches.

Ausnahmen bilden die durch den Pseudobefehl EQU (Equate) definierten Symbole. Im Falle einer EQU-Anweisung erhält das Symbol den Adreßwert des im Operandenfeld stehenden Ausdrucks und dessen Längenattribut.

Jedes Symbol, welches im Programm im Markenfeld auftritt, wird als definiert bezeichnet.

Tritt einunddasselbe Symbol ein zweites Mal im Markenfeld auf, so erfolgt bei der Übersetzung mit MBAS 1520 ein Fehlerausdruck. Gleichfalls werden nicht definierte Symbole, die im Operandenfeld auftreten, durch einen Fehlerausdruck (Anzeige) gekennzeichnet.

*Beispiel für Symbole:*

| zulässige Symbole | unzulässige Symbole |
|---|---|
| A1 | 1 |
| MARKE | MARKEX |
| X128 | 128X |
| | A+B |
| | MARK E |

## 4.4. Bezugnahme auf den Speicherplatzzuweisungszähler (*)

*Durch '*' im Operanden wird auf den aktuellen Wert des Speicherplatzzuweisungszählers bezug genommen.*

*Durch den Speicherplatzzuweisungszähler wird jeweils das linke (adreßniederwertigste) Byte des durch eine Anweisung definierten Feldes (Makrobefehl), Konstantendefinition, Speicherreservierung) adressiert.*

**Beispiel:**

*Die Anweisungen*

```
                BCT     A2, * + 5
```

*und*

```
MARKE:          BCT     A2, MARKE + 5
```

*erzeugen den gleichen Maschinencode.*

## 4.5. Dezimalzahlen (a, f, k, l, u, n, lf, q, r, nda)

*Die mit*

*a, f, k, l, n, lf, q, r, nda*

*angegebenen Dezimalzahlen werden nach vorzeichenlosen (OV) und vorzeichenhaften (VZ) unterschieden.*

| | Vorzeichen | Wertebereich |
|---|---|---|
| a | OV | $1 \leq a \leq 3$ |
| f | OV | $0 \triangleq f \leq 65535$ |
| k | OV | $k \in .(4,8).$ |
| l | OV | $1 \leq l \leq 256$ |
| u | OV | $0 \leq u \leq 255$ |
| n | OV | $0 \triangleq n \leq 15$ |
| lf | OV | $1 \leq lf \leq 255$ |
| q | VZ | $\pm (0 \triangleq q \leq 9999999)$ |
| r | VZ | $\pm (0 \leq r \leq 999\ 999\ 999\ 999\ 999)$ |
| nda | OV | $0 \triangleq nda \leq 4$ |

*Die Semantik ist der jeweiligen Befehlsbeschreibung zu entnehmen (siehe Anlage 2).*

## 4.6. Adreßdirektwert (nnnnn)

Adreßdirektwerte werden verwendet, um absolute Adressen und Verschiebungen für die Adreßrechnung anzugeben. Damit muß der Adreßdirektwert in 2 Bytes binär verschlüsselbar sein und im Wertebereich

$$\emptyset \leq nnnnn = 2^{16}-1$$

liegen.

Der Assembler akzeptiert 2 Arten von Direktwerten:

1. Dezimaler Adreßdirektwert

   $$\emptyset \leq f \leq 65535$$

   Der dezimale Adreßdirektwert wird als vorzeichenlose Zahl dargestellt.

2. Hexadezimaler Adreßdirektwert

   $$X'\emptyset\emptyset\emptyset\emptyset' \leq hx \leq X'FFFF'$$

   Der maximale Wert 65535 bzw. X'FFFF' ist die größtmögliche Hauptspeicheradresse.

## 4.7. Ausdrücke (adr), Terme (term)

Jeder Operand eines Makrobefehls oder jeder Assembleranweisung besteht aus einem oder mehreren Ausdrücken (adr). Ein Ausdruck repräsentiert stets einen Adresswert. Er besitzt folgende Form:

$$adr :: = term \quad [(\pm)\ nnnnn]$$

Treten im Operandenfeld eines Befehls mehrere Ausdrücke auf, so werden sie indexiert, z.B. adr1, adr2, adr3.

Ein Term kann aus einem Symbol bzw. einem Adreßdirektwert bzw. einer Zeichenkette des Typs H oder der Angabe des aktuellen Speicherplatzzuweisungszählers bestehen.

```
term :: =  (symbol)
           (nnnnn)
           (X'hhh')
           (*     )
```

nnnnn gibt eine absolute Verschiebung bezüglich des Terms (term) an.

### 4.7.1. Längenattribut

Ein Ausdruck kann außer seinen Adresswert noch ein Längenattribut besitzen.

Das Längenattribut spezifiziert die Länge des durch den Ausdruck adressierten Datenfeldes. Die Längenspezifikation kann explizit oder implizit erfolgen. Explizite Längenangaben dominieren über implizite.

#### 4.7.1.1. Explizite Länge

Das Längenattribut wird explizit vergeben, indem hinter dem Ausdruck die Länge als Dezimalzahl in runde Klammern eingeschlossen angegeben wird.

*Beispiel:*

```
        MVC     FELD1  +  3 (20), FELD2
```

*4.7.1.2. Implizite Länge*

*Das Längenattribut eines Ausdrucks ergibt sich aus dessen 1. Term. Dabei gelten folgende Festlegungen:*

- *Das Längenattribut eines absoluten Terms ist 1*
- *Ein Symbol im Markenfeld eines Makrobefehls hat das Längenattribut 1*
- *Bezugnahmen auf den Speicherplatzzuweisungszähler führen zum Längenattribut 1*
- *Ein EQU-Symbol erhält das Längenattribut des 1. Terms im Operanden*
- *Symbole im Markenfeld von DS- bzw. DC-Anweisungen erhalten als Längenattribut die Länge der definierten Felder bzw. Konstanten.*

*Beispiele:*

```
        MVC     FELD1 + 3, FELD2
         .
         .
         .
FELD1:  DS   20
```

*Durch den MVC-Befehl werden 20 Byte transportiert.*

```
        AP      BCD11, BCD12
         .
         .
         .
BCD11:  DS   8
BCD12:  DS   4
```

*Die Längen der durch den AP-Befehl verarbeiteten Felder betragen 8 Byte (BCD 1) und 4 Byte (BCD 2)*

```
        MVC     * + 7, WERT
```

*Es wird 1 Byte von WERT nach * + 7 transportiert.*

*4.7.2. Adreßtypen von Ausdrücken*

*Für die Ausdrücke erfolgt eine Einteilung in Adreßtypen.*

*Die Adreßtypen legen fest, ob ein Längenattribut im Ausdruck erlaubt ist bzw. welchen Wert das Längenattribut annehmen kann.*

*4.7.2.1. Adreßtyp: (at1)*

*Der Adreßtyp 1 stellt einen Ausdruck mit optionaler Längenangabe dar.*

$$at1 :: = \left\{ \begin{matrix} adr \quad (1) \\ adr \end{matrix} \right\} \qquad bzw.$$

In der Darstellung der Makrobefehlsbeschreibung

$$at\ 1 :: = adr \quad \left[(l)\right]$$

Die Indirektadressierung ist erlaubt.

Im Falle, daß

$$at1 = adr(l),$$

wird durch die Angabe von '(l)' eindeutig das Längenattribut des mit adr adressierten Bereiches vorgegeben. Für l gilt:

$$1 \leq l \leq 256$$

4.7.2.2. **Adreßtyp 2 (at2)**

Der Adreßtyp 2 läßt keine Längenangaben zu

$$at2 :: = adr$$

Indirektadressierung ist erlaubt.

4.7.2.3. **Adreßtyp 3 (at3)**

Der Adreßtyp 3 läßt Ausdrücke mit optionaler Längenangabe für gepackte BCD-Daten zu

$$at3 :: = \begin{Bmatrix} adr\ (k) \\ adr \end{Bmatrix} \quad \text{bzw. in der}$$

Darstellung in der Makrobefehlsbeschreibung

$$at3 :: = adr \quad \left[(k)\right]$$

$k \in (4,8)$ je nach Länge der definierten BCD-Felder.

4.7.2.4. **Adreßtyp 4 (at4)**

Der Adreßtyp läßt einen Ausdruck mit eingeschränkter Längenangabe zu. Die Längenangabe ist optimal.

$$at4 :: = \begin{Bmatrix} adr\ \ (l) \\ adr \end{Bmatrix}, \text{ wobei } l \leq 64$$

bzw. in der Darstellung der Makrobefehlsbeschreibung

$$at4 :: = adr \left[(l)\right], \text{ wobei } l \leq 64.$$

Indirektadressierung ist erlaubt.

4.7.2.5. **Adreßtyp 5**

Der Adreßtyp 5 läßt nur Ausdrücke ohne Längenangabe und ohne Indirektadressierung zu.

4.7.3. **Indirekte Adressierung**

Durch Einklammern der Adreßtypen at1, at2, at3 und at4 in runde Klammern "( )" ist in allen zutreffenden Befehlen eine Indirektadressierung möglich.

$$\begin{Bmatrix} (ausdruck) \\ (ausdruck\ (q)) \end{Bmatrix}$$

$$q :: = l/k$$

Die expliziten Längenangaben l und k sind in die Indirektklammern aufzunehmen.

## 4.8. Direktoperanden

Ein Direktoperand ist eine Konstante, die direkt in den Maschinencode des Befehls eingeht.

### 4.8.1. Direktoperand der Länge 1 Byte (d1)

Direktoperanden der Länge 1 Byte können sein:

$$d1 :: = \left\{ \begin{matrix} H'hh' \\ C'c' \\ at5 \end{matrix} \right\} \text{ , wobei}$$

| | |
|---|---|
| H'hh' | Zeichenketten des Typs H und der Länge 1 Byte (Beisp.: H'CF') |
| C'c' | Zeichenkette Typ C und der Länge 1 Byte (1 KOI-Zeichen) (Beisp.: C'A') |
| at5 | Ausdruck mit eingeschränktem Wertebereich $\emptyset \leq at5 \leq 255$ (z.B. EQU-Wert) |

### 4.8.2. Direktoperand der Länge 2 Byte (d2)

Direktoperanden der Länge 2 Byte können sein:

$$d2 :: = \left\{ \begin{matrix} C'cc' \\ H'hhhh' \\ at5 \end{matrix} \right\} \text{ , wobei}$$

| | |
|---|---|
| C'cc' | Zeichenkette des Typs C und der Länge 2 (Wird nur 1 Zeichen angegeben, wird das rechte Byte mit H'$\emptyset\emptyset$' aufgefüllt.) |
| H'hhhh' | Zeichenkette des Typs H und der Länge 2 Byte |
| at5 | Ausdruck |

## 4.9. Geräteadresse

$$g :: = \left\{ \begin{matrix} \emptyset \leq u \leq 255 \\ symbol \end{matrix} \right\}$$

Der Adreßwert der Geräteadresse muß zwischen $\emptyset$ und 255 liegen.

## 4.10. Sprungbedingung in Abhängigkeit von einem der Selektoren $\emptyset$ - 7 (s7)

$$s7 :: = \left\{ \begin{matrix} n \\ Nn \end{matrix} \right\} \text{ , wobei}$$

| | |
|---|---|
| n | Selektornummer $\emptyset \leq n \leq 7$ |
| N | Negation, Umkehrung der Sprungbedingung |

Beispiel: BCK 3, ZIEL

Ist Selektor 3 gesetzt, so wird zur Adresse ZIEL gesprungen.

## 4.11. Spezifikation der Selektoren im K-Register (17)

$$17 :: = \left\{ \begin{matrix} n \\ n,17 \\ \# \end{matrix} \right\} \text{ , wobei } \quad \emptyset \leq n \leq 7$$

Nach der Definition von 17 ist eine Angabe als Liste der Selektoren möglich. Die Trennung der Selektornummern erfolgt durch Komma ",".

Das Nummernzeichen (#) steht als Kurzform für alle Selektoren.

Beispiel:

1. RESK 2,7,3,1
   Die Selektoren 2,7,3 und 1 werden rückgesetzt.

2. SETK #
   Alle Selektoren werden gesetzt.

### 4.12. ON-Codes (onc)

Der ON-Code spezifiziert die Fehlerbedingung für den ON-Befehl (Verzweigung zur spezifizierten Adresse, wenn angemeldete Fehlerbedingung eingetreten).

$$onc :: = \left\{ \begin{array}{l} E \\ D \\ Z \end{array} \right\}$$

| | |
|---|---|
| E | Systemfehler mit Abbruchwirkung |
| D | Datenfehler bei BCD-Arithmetik |
| Z | Nulldivision |

### 4.13. ONF-Codes (onfc)

Der ONF-Code spezifiziert die Bedingung, unter der die durch einen ONF-Befehl angemeldete Verbindung zwischen einer Datei und einem Unterbrechungsprogramm wirksam wird

$$onfc :: = \begin{array}{l} ( \ ERR \ ) \\ ( \ EOF \ ) \\ ( \ EOV \ ) \\ ( \ MRK \ ) \\ ( \ DT \ ) \end{array}$$

| onfc | Bedeutung |
|---|---|
| ERR | Fehler (error) |
| EOF | Dateiende (end of file) |
| EOV | Datenträgerende (end of volume) |
| MRK | Markierter Satz (marked) |
| DT | Datenfernübertragung (data transmission) |

### 4.14. Positionierung bei sequentiellen Datei- E/A-Befehlen (pos)

pos dient der Positionierung von logischen Sätzen innerhalb eines sequentiell organisierten Datenbestandes.

pos :: = F / N / B / L

| pos | Bedeutung |
|---|---|
| F | first : Positionierung auf 1. Satz des Datenbestandes |
| N | next: Positionierung auf nächsten ((n+1)-ten) Satz des Datenbestandes |

| pos | Bedeutung | |
|---|---|---|
| B | back: | Positionierung auf den gerade bearbeiteten Satz des Datenbestandes |
| L | last: | Positionierung auf den letzten aufgezeichneten Satz im Datenbestand |

## 4.15. Positionierparameterliste (pparlist)

Die Angabe von pparlist erfolgt beim POS-Befehl (z.B. für Druckkopf- oder Kursorpositionierung), wobei gilt:

pparlist :: = pparlist, ppar/ppar

$$ppar :: = \left\{ \begin{array}{l} H \left\{ \begin{array}{l} + \\ - \\ = \end{array} \right\} g \\ \left\{ \begin{array}{l} V \left\{ \begin{array}{l} + \\ - \\ = \end{array} \right\} g \\ VT \end{array} \right\} \\ I = adr \\ FF \\ R \end{array} \right.$$

Koordinatenpositionierung:

| | Bedeutung |
|---|---|
| H | Horizontal |
| V | Vertikal |
| + | relativ vorwärts |
| - | relativ rückwärts |
| = | absolut |
| g | Geräteadresse |

VT - Zellensteuerfunktion
Vertikaltabulation bis Stoppunkt entsprechend CD-Tabelle (Steuertabelle), ansonsten absolute Vertikalpositionierung

FF - Neue Seite
In Abhängigkeit vom Gerät erfolgt die Ausführung. (siehe POS-Befehl)

R - Anweisen Umkehrfunktion
Diese Funktion wird gerätespezifisch ausgeführt. (siehe POS-Befehl)

I=adr - Positionierung erfolgt in Verbindung mit den Werten einer 2-Byte-Zelle, die durch einen Ausdruck adr adressiert wird.

*Beispiel:*

```
        .
        .
        .
POS     Ø,H-Ø,I = POSPA
        .
        .
DC      I'5'    ; H-Zelle
DC      I'2'    ; V-Zelle
```

*Es werden 5 Horizontalschritte rückwärts und 2 Vertikalschritte vorwärts auf dem logischen Gerät "Ø" ausgeführt.*

*Beachte:*

- *Wird H bzw. V nicht in pparlist spezifiziert, so wird vom Assembler automatisch*

  *H+Ø bzw. V+Ø*

  *generiert.*

### 4.16. Funktionscodes (fctc)

*Die nachfolgenden Funktionscode sind nur in Verbindung mit dem CTRL-Befehl zugelassen und lösen Hardwarefunktionen aus (siehe Beschreibung CTRL-Befehl).*

*In der Sprache sind für fctc zugelassen:*

*fctc :: = RUN/WTM/FOR/REW/BSR/FSR/BSF/FSF/TRP/FI/FE/BBC/RBC/*

### 4.17. Starttastencodes (stc)

*Mit Hilfe von stc lassen sich die Starttastencodes über die Tastenbezeichnung angeben.*

*stc :: = 0/1/2/3/4*
*ET1/ET2*

### 4.18. Maske (m)

*Bei den Befehlen PUTP, UNPK ist als Operand m eine Maske zu definieren.*

*m :: = [int] [deza] fs ds*

| | |
|---|---|
| *int* | *Interpunktionsangabe* |
| *deza* | *Angabe der Dezimalstellenabtrennung* |
| *fs* | *Füllnullenbehandlung* |
| *ds* | *Angabe der Zahl der abzutrennenden Dezimalstellen* |

*Interpunktionsangabe:*

*int :: = NI/HP/HS/TP/TS*

wobei

| | |
|---|---|
| NI | keine Interpunktion |
| HP | Hunderter-Punkt |
| HS | Hunderter-Space |
| TP | Tausender-Punkt |
| TS | Tausender-Space |

Angabe der Dezimalstellenabtrennung:

deza :: = C/P

wobei

| | |
|---|---|
| C | Komma |
| P | Punkt |

Füllnullenbehandlung:

fs :: = Z/S/*

| | |
|---|---|
| Z | Füllnullen werden als Ø ausgegeben |
| S | Füllnullen werden als Space ausgegeben |
| * | Füllnullen werden als * ausgegeben |

Vornullen werden stets als "Ø" ausgegeben.

Angabe der Zahl der abzutrennenden Dezimalstellen

$0 \leq ds \leq 3$

Zusammenfassung:

$$\left[\left\{\begin{matrix}NI\\HP\\HS\\TP\\TS\end{matrix}\right\}\right] \quad \left[\left\{\begin{matrix}C\\P\end{matrix}\right\}\right] \quad \left\{\begin{matrix}Z\\S\\*\end{matrix}\right\} \quad \left\{\begin{matrix}0\\1\\2\\3\end{matrix}\right\}$$

4.19. Konstantenwert (wert)

$$wert :: = \left\{\begin{matrix}F'f'\\W'q'\\D'r'\\CLI'c_1 \ldots c_1\\C'c_1 \ldots c_1\\H'h_1 \ldots h_1'\\X'hhhh'\\I'u'\\V(symbol)\\A(adr)\\Y(adr)\end{matrix}\right\}$$

wert tritt nur im Operandenfeld von Konstantenanweisungen au
und dient der Definition einer Konstanten (siehe Abschnitt
5.8.).

## 5. Beschreibung der MABS 1520 - Pseudobefehle

Makrobefehle vertreten im symbolischen Programm die Befehle, welche zum Programmausführungszeitpunkt die Arbeit des Gerätes steuern.
Die Pseudobefehle sind nur während der Übersetzung des Quellprogramms wirksam. Jedoch liefern die meisten, z.B. Pseudobefehle, die zur Konstantendefinition und Speicherplatzreservierung im Quellprogramm genutzt werden, einen echten Beitrag zum Aufbau des Phasenprogramms.

### 5.1. Beginn eines Quellprogramms

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | PN | symbol |

Diese Anweisung muß stets am Programmanfang stehen. Sie wird vom Quelltexterfassungsprogramm (Editor) automatisch generiert und kennzeichnet den Programmbeginn.

### 5.2. Ende eines Quellprogramms

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | END | keine Eintragung |

Die END-Anweisung kennzeichnet das Programmende. Sie muß unbedingt angegeben werden.

### 5.3. Seitenschaltung im Übersetzungsprotokoll

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | EJEC | keine Eintragung |

Beim Erkennen einer EJEC-Anweisung durch den Assembler, wird das Assemblerprotokoll mit einer neuen Seite fortgesetzt.

### 5.4. Zeilenvorschub im Übersetzungsprotokoll

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | SPACE | n |

Die Operanden der Anweisung anzugebende Dezimalzahl spezifiziert die Anzahl der Leerzeilen, die in das Assemblerprotokoll eingefügt werden.

$$0 \leq n \leq 15$$

## 5.5. Kennzeichnung externer Symbole

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | EXTRN | symbollist |

Die EXTRN-Anweisung kennzeichnet Symbole, die in diesem Quellmodul benutzt werden, aber in einem anderen definiert wurden (Verbindungssymbole).

symbollist : symbollist, symbol/symbol

Die Symbolliste symbollist enthält ein oder mehrere Symbole, getrennt durch Kommata. Die Kapazität der Symbolliste wird durch die Quellzeilenlänge beschränkt.
Die Symbole im Operandenfeld dürfen im gleichen Quellmodul nicht im Markenfeld einer Befehlszeile erscheinen.
Das Markenfeld darf keine Eintragung enthalten.

Beachte: Symbole aus V-Typ-Adreßkonstanten (siehe Abschnitt 5.8.) müssen nicht durch EXTRN-Anweisungen gekennzeichnet werden.

## 5.6. Definition symbolischer Eintrittspunkte

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | ENTRY | symbollist |

Eine ENTRY-Anweisung wird benutzt, um Symbole, die in einem Quellmodul definiert werden, aber von einem anderen, getrennt von ihm erfaßten, angesprochen werden sollen, als Verbindungssymbole zu kennzeichnen.

Es handelt sich dabei um Symbole, die die Verbindung von getrennt erfaßten Quellmoduln ermöglichen sollen.

symbollist :: = symbollist, symbol/symbol

Die Symbolliste symbollist enthält ein oder mehrere Symbole, getrennt durch Komma. Die Kapazität der Symbolliste wird durch die Quellzeilenlänge beschränkt.
Das Markenfeld darf keine Eintragung enthalten.
Es sind mehrere ENTRY-Anweisungen erlaubt.

### Symbolische Verbindung zwischen Quellmoduln

Für Quellmoduln, die zu einer Phase verbunden werden sollen, besteht das Problem, gewünschte Verbindungen symbolisch zu definieren. Das wird mittels der Verbindungssymbole erreicht.

Zur Definition und Kennzeichnung solcher Verbindungssymbole dienen die in diesem Abschnitt und in Abschnitt 5.5. beschriebenen Anweisungen.

*Beispiel:*

*1. Quellmodul*

```
                PN      A
                .
                .
        MARK1:  AP      FELD1,FELD2
                .
                .
                .
                MP      2,FELD1,KONX
                .
                .
                .
        MARK3:  PUTP    Ø,Z,7,FELD1
                .
                .
                .
                B       MARK2
                .
                .
                .
        FELD1:  DS      8
                ENTRY   MARK3,MARK1
                EXTRN   FELD2,KONX,MARK2
                .
                .
                .
                END
```

*2. Quellmodul*

```
                PN      B
        MARK2:  ENTP    7,FELD2
                .
                .
                .
                CP      FELD2,KONX
                BP      MARK1
                .
                .
                .
                B       MARK3
                .
                .
                .
        KONX:   DC      W'1Ø76'
        FELD2:  DS      4
                ENTRY   MARK2,FELD2,KONX
                EXTRN   MARK3,MARK1
                .
                .
                .
                END
```

## 5.7. Speicherbereichsdefinition

*Format:*

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DS | f |

*Die DS-Anweisung dient zum Reservieren und eventuell zum Benennen von Speicherbereichen.*
*f spezifiziert die Anzahl der freizuhaltenden Bytes.*

$$1 \leq f \leq 65535$$

*Der Assembler erzeugt für den freizuhaltenden Bereich keinen Inhalt, es wird nur der Speicherplatzzuweisungszähler um den Wert von f erhöht.*
*Beim Laden des Programms aus der Phasenbibliothek bleibt der (zufällige) Hauptspeicherinhalt (RAM) in diesem Bereich erhalten.*

*Bereiche, die einen definierten Anfangswert enthalten sollen, müssen mit der DC-Anweisung definiert werden bzw. sind bei Programmabarbeitung zu erzeugen.*
*Ein Symbol im Markenfeld erhält ein Adreßattribut (Wert des Speicherplatzzuweisungszählers) und ein Längenattribut (gleich f).*
*Falls f > 256, so erhält das Symbol das Längenattribut 1.*
*Bei Gebrauch des Symbols in Operanden dominiert die explizit angegebene Länge.*

*Beispiel:*

```
FELD:  DS    40   ; RESERVIERUNG 40 BYTE UNTER SYMB!
                  ; NAMEN'FELD'
```

## 5.8. Konstantendefinition

*Format:*

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | wert |

*Die DC-Anweisung dient der Definition von Konstanten im Programm. Der Assembler unterscheidet 10 Typen von Konstanten. Die im Operandenfeld angegebenen Konstanten werden im Programm abgelegt und sind damit für eine spätere Programmabarbeitung bereitgestellt.*

*Wird ein Symbol im Markenfeld der Anweisung verwendet, so besteht die Möglichkeit, in den Operanden anderer Anweisungen (Makrobefehle und Pseudobefehl EQU) auf diese Konstante symbolisch Bezug zu nehmen.*

*Ein im Markenfeld stehendes Symbol erhält ein Adreßattribut und ein Längenattribut zugeordnet.*
*Der Typ der Konstanten wird im Operanden angegeben. Es folgt in Hochkomma eingeschlossen die Konstante.*

*Mit einer Anweisung kann nur eine Konstante definiert werden.*

5.8.1. Festkommakonstante (Typ F)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | F'f' |

$$0 \leq f \leq 65535$$

Die in der Anweisung definierte Konstante wird in eine Binärzahl der Länge 2 Byte überführt und in der Reihenfolge low-Teil, high-Teil bereitgestellt.

| low-Teil | high-Teil |
|---|---|
| * | * + 1 |

* - aktueller Wert des Speicherplatzzuweisungszählers

Ein im Markenfeld angegebenes Symbol erhält das Längenattribut 2.

5.8.2. Festkommakonstante (Typ I)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | I'u' |

Es wird eine Konstante mit

$$0 \leq u \leq 255$$

definiert, die in einem Byte abgelegt wird. (Bsp.: DC I'14')

Ein Symbol im Markenfeld erhält das Längenattribut 1.

5.8.3. BCD-Konstante der Länge 4 (Typ W)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | W'q' |

Die Anweisung definiert eine BCD-Konstante (gepacktes Format) der Länge 4 Byte.

$$\pm \ ( \ 0 \leq q \leq 10^7 - 1 \ )$$

Wird kein Vorzeichen angegeben, so wird automatisch "+" angenommen. (Beispiel: DC W'-137')

Ein im Markenfeld angegebenes Symbol erhält das Längenattribut 4.

5.8.4. BCD-Konstante der Länge 8 Byte (Typ D)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | D'r' |

Die Anweisung definiert eine BCD-Konstante (gepacktes Format) der Länge 8 Byte.

$$\pm\ (\ 0 \leq r \leq 10^{15}-1)$$

Wird kein Vorzeichen angegeben, so wird automatisch "+" angenommen. (Beispiel: DC D'+1587300008169')

Ein im Markenfeld angegebenes Symbol erhält das Längenattribut 8.

5.8.5. Zeichenketten einer vereinbarten Länge l (Typ CL)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | CL1'$c_1 \ldots c_1$' |

Die Anweisung definiert eine Zeichenkette aus KOI-7-Bit-Code-Zeichen mit einer vereinbarten Länge l Bytes, wobei

$$1 \leq l \leq 256$$

(bzgl. l vgl. Abschnitt 4.6.)

Werden weniger als l vereinbarte Zeichen in der Kette angegeben, so wird die Konstante linksbündig eingetragen. Rechts wird mit Füllnullen (NUL: = H'00) aufgefüllt.

Beachte: Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellzeilenlänge beschränkt. Mit CL1'' können Speicherbereiche mit der definierten Länge l und dem Byteinhalt H'00' (gelöschte Bereiche) erzeugt werden. Werden mehr als l vereinbarte Zeichen angegeben, so wird die Konstante linksbündig eingetragen. Die überzähligen Zeichen gehen verloren.

Beachte: Besonderheit für das Zeichen ('). Dieses Zeichen ist in der Anweisung jeweils zweimal zu schreiben, wenn es in der internen Form jeweils einmal erscheinen soll.

Darstellung von nur Apostroph mit "CL1''''".
Ein im Markenfeld angegebenes Symbol erhält als Längenattribut die vereinbarte Länge l zugewiesen.

Beispiel:

| Marke: | DC | CL16'VEB ROBOTRON' |
|---|---|---|

Zeichenkette ohne Längenvereinbarung (Typ C)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | $C'c_1 \ldots c_l'$ |

Die Anweisung definiert eine Zeichenkette aus KOI-7-Bit-Code-Zeichen, d.h. sie wird wie angegeben abgelegt.

$$1 \leq l \leq 256$$

Bei Verwendung des Apostroph ist eine Besonderheit zu beachten (siehe Konstante Typ CL). Die Darstellung von nur Apostroph erfolgt mit "C''''".
Ein Symbol im Markenfeld der Anweisung erhält das Längenattribut 1.

Beachte: Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellzeilenlänge beschränkt.

5.8.6. Hexadezimalkonstante (Typ H)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | $H'h_1 \ldots h_i'$ |

Die Anweisung definiert eine Zeichenkette aus i Hexadezimalziffern, die in

$$l = INT\left(\frac{i+1}{2}\right) Bytes$$

abgelegt sind

$$0 \leq l \leq 512$$

Beachte: Durch den Editor wird die Anzahl der in der Zeichenkette angebbaren Zeichen entsprechend der Quellenzeilenlänge beschränkt.

Ein Symbol im Markenfeld erhält das Längenattribut 1.

5.8.7. Adreßkonstante – A (Typ A)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | A(adr) |

Die Anweisung definiert eine Adreßkonstante über einen absoluten oder verschieblichen Ausdruck.
Die Adreßkonstante wird in 2 Byte abgelegt.

Der Ausdruck (adr) kann aus einer beliebigen Anzahl von Termen (term) zusammengesetzt sein.

Für die A-Adreßkonstante wird das Längenattribut 2 angenommen.

5.8.8. Y-Adreßkonstante (Typ Y)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | Y(adr) |

Die Anweisung definiert eine Adreßkonstante über einen absoluten verschieblichen Ausdruck.
Der Wert der im Speicher abgelegten Konstanten entspricht dem Low-Teil des in runden Klammern angegebenen Ausdrucks.
Das Längenattribut ist 1.

Der Ausdruck (adr) kann aus einer beliebigen Zahl von Termen (term) bestehen.

5.8.9. V-Adreßkonstante (Typ V)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | V(symbol) |

Die V-Typ-Adreßkonstante definiert wie die A-Typ-Adreßkonstante einen 2-Byte Adreßwert, jedoch kann im Operanden nur ein Symbol auftreten.
Das in runden Klammern stehende Symbol wird vom Assembler als externes Symbol betrachtet.
Ein Symbol im Markenfeld erhält das Längenattribut 2.

```
Beispiel:  A1:     DC      V(WERT)
           ist analog:
           A1:     DC      A(WERT)
                   .
                   .
                   .
                   EXTRN   WERT
```

5.8.10. Adreßdirektwert der länge 2 Byte (Typ X)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | DC | X'hhhh' |

Durch die Anweisung wird ein Adreßdirektwert definiert.
Ein Symbol im Markenfeld erhält das Längenattribut 2.

5.9. Wertzuweisung für ein Symbol

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| [symbol:] | EQU | at5 |

Durch die EQU-Anweisung werden einem im Markenfeld der Anweisung stehenden Symbol Adreß- und Längenattribut des im Operan-

denfeld angegebenen Ausdrucks zugeordnet. Das Symbol übernimmt den Charakter - absolut oder verschiebbar - des Ausdrucks im Operanden.
Enthält der Ausdruck Symbole, so müssen diese alle in vorhergehenden Anweisungen des gleichen Quellprogramms definiert worden sein.

Externe Symbole sind nicht zugelassen. Als Adreßattribut erhält das Symbol den Adreßwert des Operanden. Das Längenattribut wird vom 1. Term des Operanden übernommen. Die EQU-Anweisung ermöglicht es, Ausdrücke oder Direktwerte zu benennen.

```
Beispiel:     L1:   EQU    * - 1ØØ
              KON:  EQU    258
              A:    EQU    B
```

## 5.10. Setzen des Speicherplatzzuweisungszählers

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| keine Eintragung | ORG | nnnnn |

ORG setzt den Speicherplatzzuweisungszähler auf einen gewünschten Wert. Ab ORG wird das laufende Programm bis zur END-Anweisung absolut übersetzt.
Zwischen den Anweisungen ORG und END dürfen keine Symbole im Markenfeld auftreten, welche als EXTRN bzw. ENTRY definiert sind.

Die ORG-Anweisung kann zur Definition von COMMON-Bereichen benutzt werden, indem in verschiedenen Quellprogrammen die gleiche ORG-Anweisung vor einen Datenbereich gesetzt wird.

## 5.11. Definition des statischen Dateidefinitionsblockes (DDB)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| symbol: | DDB | ddbparlist |

Die DDB-Anweisung bewirkt den Aufbau eines Dateidefinitionsblockes (statischer Dateidefinitionsblock).

```
ddbparlist :: = ddbparlist,ddbpar/ddbpar

ddbpar     :: = schlüsselwort = spezifikation
```

Die Reihenfolge der ddbpar ist beliebig.

Nicht spezifizierte Parameter (ddbpar) der ddbparlist werden standardmäßig (siehe Spalte "Standard" der DDB-Parameter-Tabelle) generiert.

Parameter dürfen nicht getrennt werden, d.h. sie müssen komplett auf einer Quellzeile stehen und werden im Falle, daß die Quellzeile für die Parameterliste nicht ausreicht in Folgezeilen eingetragen.

Ein Komma nach dem letzten Parameter jeder Zeile ist dann erforderlich. Die Folgezeile beginnt wieder mit dem Operationscode und dem Operanden.

Im Markenfeld einer Folgezeile darf kein Symbol stehen.

Beispiel:

```
DDB1:   DDB     DA = (12),
        DDB     ORG = DIR,
        DDB     NA = DATEI1,
        DDB     DT = RW, SI = COPY,
        DDB     RCS = 12Ø
```

DDB-Parameter-Tabelle

Schlüsselwort

| | | |
|---|---|---|
| DCB | Zeiger auf den dynamischen Dateidefinitionsblock | |
| | DCB = adr<br><br>keine Indirektadressierung erlaubt | Standard: automatische Generierung von DCB |
| NA | Physischer Dateiname | |
| | NA = symbol 8<br>symbol 8 :: = symbol bestehend aus max. 8 Zeichen | Standard:<br>Symbol im Markenfeld der DDB-Anweisung |
| DT | Dateityp [1] | |
| | DT = dt<br>dt :: = R/RW<br>R Lese-Datei<br>RW Lese-Schreib-Datei | Standard:<br>R |
| SI | Dateischutzindikator [1] | |
| | SI = si<br>si :: = {COPY / SAVE / DEL} | Standard:<br>COPY |

| | | |
|---|---|---|
| ORG | Organisation [1]<br>ORG = org<br>org :: = {SEQ / DIR} | Standard:<br>SEQ |
| PT | Prozedurtyp [1]<br>PT = X'hh' | keine |
| EOF | EOF-Adresse<br>EOF = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at5) | Ø<br>(Systemfehlermaß-nahme) |
| ERR | ERROR-Adresse<br>ERR = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at5) | Ø<br>(Systemfehlermaß-nahme) |
| IOA | Adresse des Pufferbereiches für einen Datenblock<br>IOA = adr<br>Beachte:<br>Indirektadressierung ist nicht erlaubt (at5) | wird in der Länge von IOS generiert |
| IOS | Länge des Pufferbereiches für einen Datenblock<br>IOS = nnnnn<br>mit Ø ≤ nnnnn ≤ 65535 | Ø |
| RCS | Maximale Satzlänge<br>RCS = adr | 128 |
| DA | Logische Geräteadressen<br>DA = (da)<br>da :: = g [,g] $^{i-1}$, wobei<br>1 ≤ i ≤ 4 | muß spezifiziert werden |

Anmerkung:
1) PT und [SI,DT,ORG] sind alternativ anzugeben. PT gibt den Prozedurtyp für die OFÜ an, während SI, DT, ORG die Dateiarbeit mit magnetischen Datenträgern charakterisieren.

## 5.12. Definition des dynamischen Dateldefinitionsblockes (DCB)

Format:

| Markenfeld | OP-Codefeld | Operandenfeld |
|---|---|---|
| symbol: | DCB | nda |

Die Pseudoanweisung DCB reserviert 13+ndax13Bytes, wobei nda die Anzahl der Geräteadressen ist.

$$0 \leq nda \leq 4$$

Anlage 1:

Gruppeneinteilung der Makrobefehle nach der Operandenstruktur

| | | | |
|---|---|---|---|
| 1. | OPC | | RET, SOP, EOP, NOP, EOI, EXIT, BELL |
| 2. | OPC | at2 | B, BZ, BM, BP, BO, BNZ, BNM, BNP, BNO, BE, BL, BH, BNE, BNL, BNH, CALL, CALC, OPEN, CLOSE |
| 3. | OPC | at1,at2 | MVC, N, O, X, TR, CLC |
| 4. | OPC | at2,at2 | AB, SB, MB, CB, BCT, EX |
| 5. | OPC | at2,at3 | CVB |
| 6. | OPC | at3,at2 | CVD |
| 7. | OPC | at3,at3 | AP, APR, SP, SPR, CP |
| 8. | OPC | at3,at4 | PACK |
| 9. | OPC | at2,d1 | MVI, CLI, TM |
| 10. | OPC | at2,d2 | MVBI, ABI, CBI |
| 11. | OPC | d1 | SETPS |
| 12. | OPC | 17 | SETK, RESK |
| 13. | OPC | n,at2 | BC |
| 14. | OPC | n,at3 | SL, SR, ENTP |
| 15. | OPC | g,at1 | GET, PUTC |
| 16. | OPC | l,at2 | PA, PL |
| 17. | OPC | lf,at1 | ENTF, ENTU |
| 18. | OPC | s7,at2 | BCS, BCK |
| 19. | OPC | onc,at2 | ON |
| 20. | OPC | stc,at2 | BCN, BCA |
| 21. | OPC | n,at3,at3 | MP, DP |
| 22. | OPC | m,at4,at3 | UNPK |
| 23. | OPC | onfc,at2,at2 | ONF |
| 24. | OPC | at2,at2,pos | READ, WRITE |
| 25. | OPC | at2,at2,at2 | READD, WRITD |

| | | | |
|---|---|---|---|
| 26. | OPC | at2,fctc | CTRL |
| 27. | OPC | g,lf,at1 | PUTF |
| 28. | OPC | g,m,n,at3 | PUTP |
| 29. | OPC | g [(a)] [,pparlist] | POS |
| 30. | OPC | g,at2 | CD |

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| N | Logisches "UND" | N adr1 [(1)] ,adr2 | 2 | 19 l ad r1 ad r2 |
| O | Logisches "ODER" | O adr1 [(1)] ,adr2 | 2 | 1A l ad r1 ad r2 |
| X | Logisches "Exclusive ODER" | X adr1 [(1)] ,adr2 | 2 | 1B l ad r1 ad r2 |
| MVC | Transport rechts beginnend | MVC adr1 [(1)] ,adr2 | 2 | 10 l ad r1 ad r2 |
| MVI | Transport Direktoperand | MVI adr,d1 | 1 | 60 d1 ad r |
| CLC | Logischer Vergleich links beginnend | CLC adr1 [(1)] ,adr2 | 2 | 1C l ad r1 ad r2 |
| CLI | Vergleich mit Direktoperand | CLI adr1,d1 | 1 | 61 d1 ad r |
| TM | Prüfen der Bitkonfiguration eines Bytes | TM adr1,d1 | 1 | 62 d1 ad r |
| AB | Binäre Addition | AB adr1,adr2 | 2 | 30 ad r1 ad r2 |
| SB | Binäre Subtraktion | SB adr1,adr2 | 2 | 31 ad r1 ad r2 |
| MB | Binäre Multiplikation | MB adr1,adr2 | 2 | 32 ad r2 ad r2 |
| ABI | Addition eines 2-Byte-Direktop. | ABI adr,d2 | 1 | 65 d2 ad r |
| MVBI | Transport eines 2-Byte-Direktop. | MVBI adr,d2 | 1 | 64 d2 ad r |
| CB | Binärer Vergleich | CB adr1,adr2 | 2 | 33 ad r1 ad r2 |

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| CBI | Binärer Vergleich mit 2-Byte-Direkt-operanden | CBI adr,d2 | 1 | 63 d2 ad r |
| TR | Codewandlung | TR adr1 [(1)] ,adr2 | 2 | 18 1 ad r1 ad r2 |
| B | Unbedinger Sprung | B adr | 1 | 0D F0 ad r |
| BC | Bedingter Sprung in Abh. vom Bedingungscode | BC n,adr | 1 | 0D n 0 ad r |
| BCA | in Abh. vom Starttastencode Reg.A | BCA stc,adr | 1 | 03 stc ad r |
| BCN | in Abh. vom Starttastencode Reg.N | BCN stc,adr | 1 | 04 stc ad r |
| BCK | in Abh. von Sel.-register K | BCK s7,adr | 1 | 06 sc ad r |
| BCS | in Abh. von Statusreg. S | BCS s7,adr | 1 | 05 sc ad r |
| BCT | in Abh. von einem Zählwert | BCT adr1,adr2 | 2 | 3D ad r1 ad r2 |
| ON | in Abh. des ON-Codes | ON onc,adr | 1 | 08 onc ad r |
| EXIT | Aufheben des Unterbrechungsstatus | EXIT | 0 | CB |
| CALL | Aufruf eines UP's in Makrobefehlssprache | CALL adr | 1 | 0B ad r |
| CALC | Aufruf eines UP's in Mikrobefehlssprache | CALC adr | 1 | 0C ad r |

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| RET | Rückkehr vom UP | RET | 0 | C1 |
| PA | Laden der Adr. der Parameterliste in das UP | PA l, adr | 1 | 09 l adr |
| PL | Laden Parameterliste in das UP | PL l,adr | 1 | 0A l adr |
| EOP | Programmende | EOP | 0 | C2 |
| SOP | Start eines Makroprogrammes | SOP | 0 | EF |
| EOI | Aufheben Interpreterstatus | EOI | 0 | C0 |
| NOP | keine Operation | NOP | 0 | 00 |
| SETK | Setzen Selektoren Gruppe K | SETK 17 | 0 | C6 l 15 |
| RESK | Rücksetzen Selektoren Gruppe K | RESK 17 | 0 | C7 l 15 |
| EX | Befehlsaustausch mit Modifizierung | EX adr1,adr2 | 2 | 3E adr1 adr2 |
| SETPS | Setzen des Programmstatus | SETPS d1 | 0 | CA d1 |
| ENTF | Mehrzeilige Zeicheneingabe | ENTF lf,adr [(1)] | 1 | 23 l adr lf |
| PUTC | Zeichenausgabe | PUTC g,adr [(1)] | 1 | 2A l adr g |
| PUTF | Mehrzeilige Zeichenausgabe | PUTF g,lf,adr [(1)] | 1 | 28 l adr g lf |
| GET | Eingabe vom logischen Gerät | GET g,adr [(1)] | 1 | 20 l adr g |

| | Bezeichnung | Assemblerschreibweise | Anzahl I-Bit | Codierung |
|---|---|---|---|---|
| POS | Positionierung Drucker bzw. Bildschirm | POS g [(a)] [,pparlist] | 0 | 2C \| par \| H \| V \| g \| |
| CD | Definition einer Steuertabelle | CD g,adr | 1 | 27 \| par \| ad \| r \| g \| |
| CTRL | Steuerung peripherer Geräte | CTRL adr,fctc | 1 | 2E \| fctc \| ad \| r \| |
| OPEN | Eröffnen einer Datei | OPEN adr | 1 | 68 \| ad \| r \| |
| CLOSE | Abschließen einer Datei | CLOSE adr | 1 | 6C \| ad \| r \| |
| READ | Lesen eines Satzes vom Datenträger | READ adr1,adr2,pos | $2_z$ | 40 \| z2 \| ad \| r1 \| ad \| r2 \| |
| WRITE | Schreiben eines Satzes auf den DT | WRITE adr1,adr2,pos | $2_z$ | 44 \| z2 \| ad \| r1 \| ad \| r2 \| |
| READD | Lesen eines direkt adressierten Satzes | READD adr1,adr2,adr3 | $3_z$ | 41 \| z3 \| ad \| r1 \| ad \| r2 \| ad \| r3 |
| WRITD | Schreiben eines direkt adressierten Satzes | WRITD adr1,adr2,adr3 | $3_z$ | 45 \| z3 \| ad \| r1 \| ad \| r2 \| ad \| r3 |
| ONF | Herstellen einer Verbindung zwischen UP und Datei | ONF onfc, adr1,adr2 | $2_z$ | 47 \| z3 \| ad \| r1 \| ad \| r2 \| |
| AP | Addition gepackt | AP adr1 [(K1)],adr2 [(K2)] | 2 | 3A \| p1 \| ad \| r1 \| ad \| r2 \| |
| APR | Subtraktion reverse | APR adr1 [(K1)],adr2 [(K2)] | 2 | 3A \| p2 \| ad \| r1 \| ad \| r2 \| |
| SP | Subtraktion gepackt | SP adr1 [(K1)],adr2 [(K2)] | 2 | 3A \| p3 \| ad \| r1 \| ad \| r2 \| |
| SPR | Subtraktion reverse | SPR adr1 [(K1)],adr2 [(K2)] | 2 | 3A \| p4 \| ad \| r1 \| ad \| r2 \| |

| | Bezeichnung | Assemblerschreibweise | | Anzahl I-Bit | Codierung |
|---|---|---|---|---|---|
| MP | Multiplikation gepackt | MP | n,adr1 [(K1)] ,adr2 [(K2)] | 2 | 3A \| p5 \| ad \| r1 \| ad \| r2 \| |
| DP | Division gepackt | DP | n,adr1 [(K1)] ,adr2 [(K2)] | 2 | 3A \| p6 \| ad \| r1 \| ad \| r2 \| |
| CP | Vorzeichengerechter Vergleich gepackt | CP | adr1 [(K1)], adr2 [(K2)] | 2 | 3A \| p7 \| ad \| r1 \| ad \| r2 \| |
| SL | Linksverschiebung gepackt | SL | n,adr [(K)] | 1 | 3C \| p8 \| ad \| r \| |
| SR | Rechtsverschiebung gepackt | SR | n,adr [(K)] | 1 | 3C \| p8 \| ad \| r \| |
| ENTP | Eingabe gepackt | ENTP | n,adr [(K)] | 1 | 21 \| p10 \| ad \| r \| |
| PUTP | Ausgabe gepackt mit Markenaufbereitung | PUTP | g,m,n,adr [(K)] | | 29 \| p10 \| ad \| r \| g \| m \| |
| CVB | Konvertieren gepackt in binär | CVB | adr1,adr2 [(K)] | 2 | 38 \| p11 \| adr \| 1 \| adr \| 2 \| |
| CVD | Konvertieren binär in gepackt | CVD | adr1 [(K)] ,adr2 | 2 | 39 \| p11 \| adr \| 1 \| adr \| 2 \| |
| PACK | Packen ungepackte Dezimalzahl | PACK | adr1 [(K)] ,adr2 [(1)] | 2 | 37 \| p12 \| adr \| 1 \| adr \| 2 \| |
| UNPK | Entpacken gepackte | UNPK | m,adr1 [(1)],adr2 [(K)] | 2 | 36 \| p12 \| adr \| 1 \| adr \| 2 \| m |

*Anlage 2*

**Pseudobefehle**

| | | | |
|---|---|---|---|
| Beginn Quellprogramm | | PN | symbol |
| Ende Quellprogramm | | END | |
| Seitenschaltung Übersetzungsprotokoll | | EJEC | |
| Zeilenvorschub Übersetzungsprotokoll | | SPACE | n |
| Kennzeichnung externer Symbole | | EXTRN | symbollist |
| Difinition symbolischer Eintrittspunkte | | ENTRY | symbollist |
| Speicherbereichsdefinition | [symbol:] | DS | f |
| Konstantendefinition | [symbol:] | DC | wert |
| . Typ F | [symbol:] | DC | F'f' |
| . Typ W | [symbol:] | DC | W'q' |
| . Typ D | [symbol:] | DC | D'r' |
| . Typ CL | [symbol:] | DC | CL1'$c_1 \ldots c_l$' |
| . Typ C | [symbol:] | DC | C'$c_1 \ldots c_l$ |
| . Typ H | [symbol:] | DC | H'$h_1 \ldots h_i$ |
| . Typ A | [symbol:] | DC | A(adr) |
| . Typ X | [symbol:] | DC | X'$h_1 h_2 h_3 h_4$' |
| . Typ I | [symbol:] | DC | I'U' |
| . Typ V | [symbol:] | DC | V(symbol) |
| . Typ Y | [symbol:] | DC | Y(adr) |
| Wertzuweisung für ein Symbol | [symbol:] | EQU | adr |
| Setzen Speicherplatzzuweisungszähler | | ORG | nnnnn |
| Definition des statischen Dateidefinitionsblockes | symbol:: | DDB | ddbparlist |
| | ddbparlist :: = ddbparlist, ddbpar/ ddbpar | | |
| | ddbpar :: = Schlüsselwort-spezifikation | | |
| Definition des dynamischen Dateidefinitionsblockes | symbol: | DCB | nda |

| | | |
|---|---|---|
| Adreßdirektwert | nnnnn | nnnnn ::= (0 ≤ f ≤ 65535 )<br>(X'0000' ≤ hx ≤ X'FFFF') |
| Ausdrücke | adr | adr::= term ± nnnnn |
| Term | term | term :: = ( symbol )<br>( nnnnn )<br>( X'hhhh')<br>( * ) |
| Direktoperand der Länge 1 Byte | d1 | d1 :: = ( C'c' )<br>( H'hh' )<br>( at5 ) |
| Direktoperand der Länge 2 Byte | d2 | d2 :: = ( C'cc'<br>( H'hhhh')<br>( at5 ) |
| Absoluter Ausdruck mit eingeschränktem Wert | g | g :: = ( u )<br>( symbol ) |
| Sprungbedingung in Abhängigkeit von Selektoren | s7 | s7 :: = ( n ) N Negation<br>( Nn ) 0 ≤ n ≤ 7 |
| Bitpositionen des K-, S-, KR-Registers | 17 | 17 :: = ( n )<br>( n,17 ) 0 ≤ n ≤ 7 |
| ON-Codes | onc | onc :: = ( E ) ERROR<br>( D ) DATA<br>( Z ) ZERO-DIVISION |
| Alpha-numerische Zeichen | c | c ∈ (A,...Z,0,1,...9,@,',...) |

| | | | |
|---|---|---|---|
| Hexadezimalziffern | h | h ∈ (Ø,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) | |
| Symbol | symbol | 1...5 Zeichen, 1. Zeichen = Buchstabe, keine Sonderzeichen kein Leerzeichen (Ø)! | |
| Bezugnahme auf Speicherplatzzuweisungszähler | * | | |
| Dezimalzahlen | a | 1 ≦ a ≦ 3 | Untergeräteadresse |
| | f | Ø ≦ f ≦ 65535 | Binärkonstante |
| | k | k ∈ (4,8) | Längenattribut BCD |
| | l | 1 ≦ l ≦ 256 | Explizite Länge |
| | u | Ø ≦ u ≦ 255 | Geräteadresse absolut |
| | n | Ø ≦ n ≦ 15 | Eingabestellen |
| | lf | 1 ≦ lf ≦ 255 | Zeilenanzahl mehrzeilig |
| | q | ± (Ø ≦ q ≦ $10^7$-1) | Wertebereich BCD |
| | r | ± (Ø ≦ r ≦ $10^{15}$-1) | Wertebereich BCD |
| | nda | Ø ≦ nda ≦ 4 | Anzahl der Geräteadressen DCB |
| ONF-Codes | onfc | onfd :: = ( ERR ) ERROR<br>( EOF ) END OF FILE<br>( EOV ) END OF VOLUME<br>( MRK ) MARKED<br>( DT ) DATA TRANSMISSION | |
| Positionierung bei sequentiellen Datei-E/A-Befehlen | pos | pos :: = ( F ) FIRST<br>( N ) NEXT<br>( B ) BACK<br>( L ) LAST | |

| | | |
|---|---|---|
| Positionierparameter-liste | pparlist | pparlist :: = pparlist, ppar/ppar<br>ppar :: = ( H ( + / − / = ) g / V ( + / − / = ) g / VT / I = adr / FF / R ) |
| Funktionscodes | fctc | fctc :: = RUN/WTM/FOR/REW/BSR/FSR/BSF/FSF/TRP/RBC/EBC/FI/FE |
| Starttastencodes | stc | stc :: = Ø/1/2/3/4/ET1/ET2 |
| Maske | m | m :: = [int] [deza] fs ds<br>int :: = NI/HP/HS/TP/TS<br>deza :: = C/P<br>fs :: = Z/S/*<br>Ø = ds = 3<br>m :: = [NI/HP/HS/TP/TS] [C/P] Z/S/* Ø/1/2/3 |

| | | |
|---|---|---|
| Konstantenwert | wert | WERT :: = ( F'f' ) ( W'q' ) ( D'r' ) ( CL1'$c_1 \ldots c_l$' ) ( C'$c_1 \ldots c_l$' ) ( H'$h_1 \ldots h_t$' ) ( X'hhhh' ) ( I'u' ) ( V(symbol) ) ( A(adr) ) ( Y(adr) ) |
| Symbolliste | symbollist | symbollist :: = symbollist, symbol/symbol |
| DDB-Parameter | ddbpar | DCB = adr<br>NA = (symbol 8)<br>DT = dt dt :: = R/RW<br>SI = si si :: = (COPY) (SAVE) (DEL )<br>ORG = org org::=(SEQ) (DIR)<br>PT = x'hh'<br>EOF = at5<br>ERR = at5<br>IOA = at5<br>IOS = nnnnn<br>RCS = nnnnn<br>DA = (lda) lda :: = g [,g] $*^{i-1}$; $1 \leq i \leq 4$ |

robotron

**VEB Robotron**
**Büromaschinenwerk Sömmerda**

DDR - 523 Sömmerda
Weißenseer Straße 52

robotron

**Robotron Export-Import**

Volkseigener
Außenhandelsbetrieb der
Deutschen Demokratischen
Republik
DDR - 108 Berlin
Friedrichstraße 61

Rl 427/85 IV/1/19